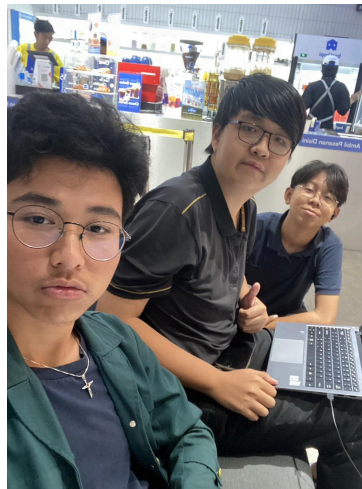


# LAPORAN TUGAS BESAR 1

ALJABAR LINEAR DAN GEOMETRI (IF2123)

*Sistem Persamaan Linier, Determinan, dan Aplikasinya*

Dosen: Dr. Judhi Santoso, M.Sc., Arrival Dwi Sentosa, S.Kom., M.T.



Disusun Oleh Kelompok Justice League

Jonathan Levi	13523132
Sebastian Enrico Nathanael	13523134
Jonathan Kenan Budianto	13523139

**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
BANDUNG  
OKTOBER 2024**

# BAB 1

## DESKRIPSI MASALAH

### 1.1 Sistem Persamaan Linier (SPL)

Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. Andstrea sudah mempelajari berbagai metode untuk menyelesaikan SPL, termasuk menghitung determinan matriks. Sembarang SPL dapat diselesaikan dengan beberapa metode, yaitu metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan ( $x = A^{-1}b$ ), dan kaidah *Cramer* (khusus untuk SPL dengan  $n$  peubah dan  $n$  persamaan). Solusi sebuah SPL mungkin tidak ada, banyak (tidak berhingga), atau hanya satu (unik/tunggal).

$$\begin{bmatrix} 0 & \mathbf{2} & 1 & -1 \\ 0 & 0 & \mathbf{3} & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & \mathbf{1} & 0 & -\frac{2}{3} \\ 0 & 0 & \mathbf{1} & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Gambar 1. Eliminasi Gauss dilakukan dengan matriks eselon baris dan eliminasi Gauss-Jordan dengan matriks eselon baris tereduksi.

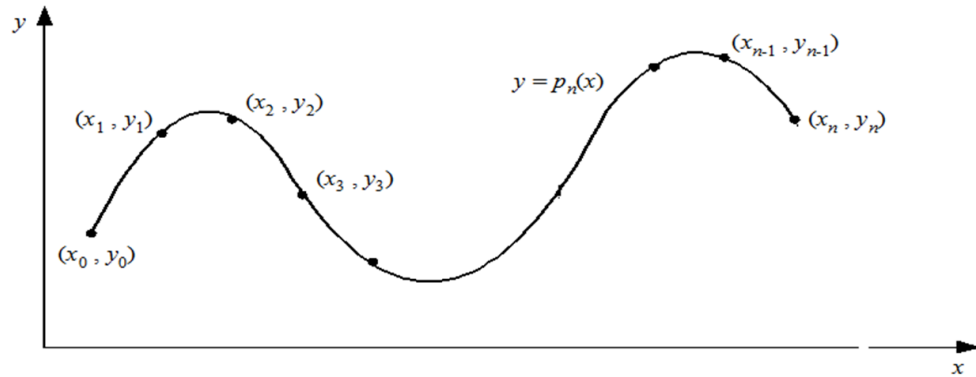
Di dalam Tugas Besar 1 ini, Anda diminta membuat satu atau lebih *library* aljabar linier dalam Bahasa Java. Library tersebut berisi fungsi-fungsi seperti eliminasi Gauss, eliminasi Gauss-Jordan, menentukan balikan matriks, menghitung determinan, kaidah *Cramer* (kaidah *Cramer* khusus untuk SPL dengan  $n$  peubah dan  $n$  persamaan). Selanjutnya, gunakan *library* tersebut di dalam program Java untuk menyelesaikan berbagai persoalan yang dimodelkan dalam bentuk SPL, menyelesaikan persoalan interpolasi, dan persoalan regresi. Penjelasan tentang interpolasi dan regresi adalah seperti di bawah ini.

Beberapa tulisan cara membuat library di Java:

1. <https://www.programcreek.com/2011/07/build-a-java-library-for-yourself/>
2. <https://developer.ibm.com/tutorials/j-javalibrary/>

### 1.2 Interpolasi Polinomial

Persoalan interpolasi polinom adalah sebagai berikut: Diberikan  $n+1$  buah titik berbeda,  $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ . Tentukan polinom  $p_n(x)$  yang menginterpolasi (melewati) semua titik-titik tersebut sedemikian rupa sehingga  $y_i = p_n(x_i)$  untuk  $i = 0, 1, 2, \dots, n$ .



Gambar 2. Ilustrasi beberapa titik yang diinterpolasi secara polinomial.

Setelah polinom interpolasi  $p_n(x)$  ditemukan,  $p_n(x)$  dapat digunakan untuk menghitung perkiraan nilai  $y$  di sembarang titik di dalam selang  $[x_0, x_n]$ .

Polinom interpolasi derajat  $n$  yang menginterpolasi titik-titik  $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$  adalah berbentuk  $p_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ . Jika hanya ada dua titik,  $(x_0, y_0)$  dan  $(x_1, y_1)$ , maka polinom yang menginterpolasi kedua titik tersebut adalah  $p_1(x) = a_0 + a_1x$  yaitu berupa persamaan garis lurus. Jika tersedia tiga titik,  $(x_0, y_0), (x_1, y_1)$ , dan  $(x_2, y_2)$ , maka polinom yang menginterpolasi ketiga titik tersebut adalah  $p_2(x) = a_0 + a_1x + a_2x^2$  atau persamaan kuadrat dan kurvanya berupa parabola. Jika tersedia empat titik,  $(x_0, y_0), (x_1, y_1), (x_2, y_2)$ , dan  $(x_3, y_3)$ , polinom yang menginterpolasi keempat titik tersebut adalah  $p_3(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ , demikian seterusnya. Dengan cara yang sama kita dapat membuat polinom interpolasi berderajat  $n$  untuk  $n$  yang lebih tinggi asalkan tersedia  $(n+1)$  buah titik data. Dengan menyulihkan  $(x_i, y_i)$  ke dalam persamaan polinom  $p_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$  untuk  $i = 0, 1, 2, \dots, n$ , akan diperoleh  $n$  buah sistem persamaan linier dalam  $a_0, a_1, a_2, \dots, a_n$ ,

$$\begin{aligned} a_0 + a_1x_0 + a_2x_0^2 + \dots + a_nx_0^n &= y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + \dots + a_nx_1^n &= y_1 \\ &\dots \\ a_0 + a_1x_n + a_2x_n^2 + \dots + a_nx_n^n &= y_n \end{aligned}$$

Solusi sistem persamaan linier ini, yaitu nilai  $a_0, a_1, \dots, a_n$ , diperoleh dengan menggunakan metode eliminasi Gauss yang sudah anda pelajari. Sebagai contoh, misalkan diberikan tiga buah titik yaitu  $(8.0, 2.0794)$ ,  $(9.0, 2.1972)$ , dan  $(9.5, 2.2513)$ . Tentukan polinom interpolasi kuadratik lalu estimasi nilai fungsi pada  $x = 9.2$ . Polinom kuadratik berbentuk  $p_2(x) = a_0 + a_1x + a_2x^2$ . Dengan menyulihkan ketiga buah titik data ke dalam polinom tersebut, diperoleh sistem persamaan linier yang terbentuk adalah

$$\begin{aligned} a_0 + 8.0a_1 + 64.00a_2 &= 2.0794 \\ a_0 + 9.0a_1 + 81.00a_2 &= 2.1972 \\ a_0 + 9.5a_1 + 90.25a_2 &= 2.2513 \end{aligned}$$

Penyelesaian sistem persamaan dengan metode eliminasi Gauss menghasilkan  $a_0 = 0.6762$ ,  $a_1 = 0.2266$ , dan  $a_2 = -0.0064$ . Polinom interpolasi yang melalui ketiga buah titik tersebut adalah  $p_2(x) = 0.6762 + 0.2266x - 0.0064x^2$ . Dengan menggunakan polinom ini, maka

nilai fungsi pada  $x = 9.2$  dapat ditaksir sebagai berikut:  $p_2(9.2) = 0.6762 + 0.2266(9.2) - 0.0064(9.2)^2 = 2.2192$ .

### 1.3 Regresi Berganda

Regresi (akan dipelajari lebih lanjut di Probabilitas dan Statistika) merupakan salah satu metode untuk memprediksi nilai selain menggunakan Interpolasi Polinom. Pada tugas besar ini, anda diminta untuk membuat 2 jenis regresi yaitu Regresi Linier Berganda dan Regresi Kuadratik Berganda.

#### 1. Regresi Linier Berganda

Meskipun sudah ada persamaan jadi untuk menghitung regresi linear sederhana, terdapat persamaan umum dari regresi linear yang bisa digunakan untuk regresi linear berganda, yaitu.

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \epsilon_i$$

Untuk mendapatkan nilai dari setiap  $\beta_i$  dapat digunakan *Normal Estimation Equation for Multiple Linear Regression* sebagai berikut:

$$\begin{array}{ccccccc} nb_0 + b_1 \sum_{i=1}^n x_{1i} & + & b_2 \sum_{i=1}^n x_{2i} & + & \cdots & + & b_k \sum_{i=1}^n x_{ki} & = & \sum_{i=1}^n y_i \\ b_0 \sum_{i=1}^n x_{1i} + b_1 \sum_{i=1}^n x_{1i}^2 & + & b_2 \sum_{i=1}^n x_{1i}x_{2i} & + & \cdots & + & b_k \sum_{i=1}^n x_{1i}x_{ki} & = & \sum_{i=1}^n x_{1i}y_i \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ b_0 \sum_{i=1}^n x_{ki} + b_1 \sum_{i=1}^n x_{ki}x_{1i} & + & b_2 \sum_{i=1}^n x_{ki}x_{2i} & + & \cdots & + & b_k \sum_{i=1}^n x_{ki}^2 & = & \sum_{i=1}^n x_{ki}y_i \end{array}$$

#### 2. Regresi Kuadratik Berganda

Dalam kasus ini, proses mengubah data-data dalam regresi kuadratik berganda cukup berbeda dengan Regresi Linier Berganda. Bentuk persamaan dari regresi kuadratik ada 3, yaitu:

- Variabel Linier: Variabel dengan derajat satu seperti X, Y, dan Z
- Variabel Kuadrat: Variabel dengan derajat dua seperti  $X^2$
- Variabel Interaksi: 2 Variabel dengan derajat satu yang dikalikan dengan satu sama lain seperti XY, YZ, dan XZ

Setiap n-peubah, jumlah variabel linier, kuadrat, dan interaksi akan berbeda-beda. Perhatikan contoh regresi kuadratik 2 variabel peubah sebagai berikut!

$$\begin{pmatrix} N & \sum u_i & \sum v_i & \sum u_i^2 & \sum u_i v_i & \sum v_i^2 \\ \sum u_i & \sum u_i^2 & \sum u_i v_i & \sum u_i^3 & \sum u_i^2 v_i & \sum u_i v_i^2 \\ \sum v_i & \sum u_i v_i & \sum v_i^2 & \sum u_i^2 v_i & \sum u_i v_i^2 & \sum v_i^3 \\ \sum u_i^2 & \sum u_i^3 & \sum u_i^2 v_i & \sum u_i^4 & \sum u_i^3 v_i & \sum u_i^2 v_i^2 \\ \sum u_i v_i & \sum u_i^2 v_i & \sum u_i v_i^2 & \sum u_i^3 v_i & \sum u_i^2 v_i^2 & \sum u_i v_i^3 \\ \sum v_i^2 & \sum u_i v_i^2 & \sum v_i^3 & \sum u_i^2 v_i^2 & \sum u_i v_i^3 & \sum v_i^4 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum y_i u_i \\ \sum y_i v_i \\ \sum y_i u_i^2 \\ \sum y_i u_i v_i \\ \sum y_i v_i^2 \end{pmatrix}$$

N menandakan jumlah peubah, terdapat 2 variabel linier yaitu  $u_i$  dan  $v_i$ , 2 variabel kuadrat yaitu  $u_i^2$  dan  $v_i^2$ , dan 1 variabel interaksi yaitu  $uv$ . Untuk setiap n-peubah, akan terdapat 1 konstan N (Terlihat di bagian atas kiri gambar), n variabel linier, n variabel kuadrat, dan  $C_2^n$  variabel linier (dengan syarat  $n > 1$ ). Tentu dengan bertambahnya peubah n, ukuran matriks akan bertambah lebih besar dibandingkan regresi linier berganda tetapi solusi tetap bisa didapat dengan menggunakan SPL.

Kedua model regresi yang dijadikan sistem persamaan linier tersebut diselesaikan dengan menggunakan metode eliminasi Gauss.

#### 1.4 Bicubic Spline Interpolation

*Bicubic spline interpolation* adalah metode interpolasi yang digunakan untuk mengaproksimasi fungsi di antara titik-titik data yang diketahui. *Bicubic spline interpolation* melibatkan konsep *spline* dan konstruksi serangkaian polinomial kubik di dalam setiap sel segi empat dari data yang diberikan. Pendekatan ini menciptakan permukaan yang halus dan kontinu, memungkinkan untuk perluasan data secara visual yang lebih akurat daripada metode interpolasi linear.

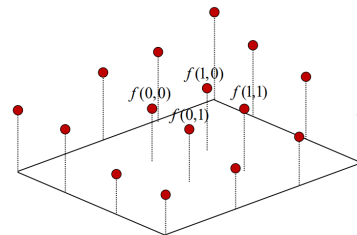
Dalam pemrosesan menggunakan interpolasi *bicubic spline* digunakan 16 buah titik, 4 titik referensi utama di bagian pusat, dan 12 titik di sekitarnya sebagai aproksimasi turunan dari keempat titik referensi untuk membangun permukaan bikubik. Bentuk pemodelannya adalah sebagai berikut.

Normalization:  $f(0,0), f(1,0)$

$f(0,1), f(1,1)$

Model:  $f(x,y) = \sum_{j=0}^3 \sum_{i=0}^3 a_{ij} x^i y^j$

Solve:  $a_{ij}$



Gambar 3. Pemodelan interpolasi *bicubic spline*.

Selain melibatkan model dasar, juga digunakan model turunan berarah dari kedua sumbu, baik terhadap sumbu  $x$ , sumbu  $y$ , maupun keduanya. Persamaan polinomial yang digunakan adalah sebagai berikut.

$$f(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j$$

$$f_x(x, y) = \sum_{j=0}^3 \sum_{i=1}^3 a_{ij} i x^{i-1} y^j$$

$$f_y(x, y) = \sum_{j=1}^3 \sum_{i=0}^3 a_{ij} j x^i y^{j-1}$$

$$f_{xy}(x, y) = \sum_{j=0}^3 \sum_{i=0}^3 a_{ij} i j x^{i-1} y^{j-1}$$

Dengan menggunakan nilai fungsi dan turunan berarah tersebut, dapat terbentuk sebuah matriks solusi  $X$  yang membentuk persamaan penyelesaian sebagai berikut.

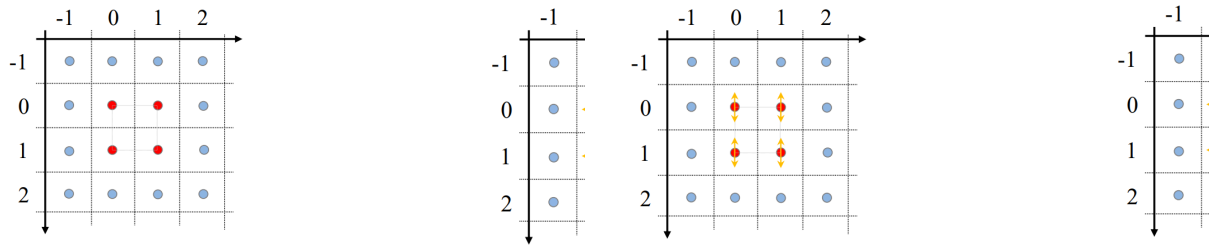
$$y = Xa$$

$$\begin{bmatrix} f(0,0) \\ f(1,0) \\ f(0,1) \\ f(1,1) \\ f_x(0,0) \\ f_x(1,0) \\ f_x(0,1) \\ f_x(1,1) \\ f_y(0,0) \\ f_y(1,0) \\ f_y(0,1) \\ f_y(1,1) \\ f_{xy}(0,0) \\ f_{xy}(1,0) \\ f_{xy}(0,1) \\ f_{xy}(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 2 & 4 & 6 & 0 & 3 & 6 & 9 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \\ a_{01} \\ a_{11} \\ a_{21} \\ a_{31} \\ a_{02} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{03} \\ a_{13} \\ a_{23} \\ a_{33} \end{bmatrix}$$

Perlu diketahui bahwa elemen pada matriks  $X$  adalah nilai dari setiap komponen koefisien  $a_{ij}$  yang diperoleh dari persamaan fungsi maupun persamaan turunan yang telah dijelaskan sebelumnya. Sebagai contoh, elemen matriks  $X$  pada baris 8 kolom ke 2 adalah koefisien dari  $a_{10}$  pada ekspansi sigma untuk  $f_x(1, 1)$  sehingga diperoleh nilai konstanta  $1 \times 1^{1-1} \times 1^0 = 1$ , sesuai dengan isi matriks  $X$ .

Nilai dari vektor  $a$  dapat dicari dari persamaan  $y = Xa$ , lalu vektor  $a$  tersebut digunakan sebagai nilai variabel dalam  $f(x, y)$ , sehingga terbentuk fungsi interpolasi bicubic sesuai model. Tugas Anda pada studi kasus ini adalah membangun persamaan  $f(x, y)$  yang akan digunakan untuk melakukan interpolasi berdasarkan nilai  $f(a, b)$  dari masukan matriks 4 x 4. Nilai masukan

$a$  dan  $b$  berada dalam rentang  $[0, 1]$ . Nilai yang akan diinterpolasi dan turunan berarah disekitarnya dapat diilustrasikan pada titik berwarna merah pada gambar di bawah.



Gambar 4. Nilai fungsi yang akan di interpolasi pada titik merah, turunan berarah terhadap sumbu  $x$ , terhadap sumbu  $y$ , dan keduanya (kiri ke kanan).

Untuk studi kasus ini, buatlah matriks  $X$  menggunakan persamaan yang ada (tidak *hardcode*) serta carilah invers matriks  $X$  dengan *library* yang telah kalian buat dalam penyelesaian masalah. Berikut adalah sebuah tautan yang dapat dijadikan referensi..

## BAB 2

### TEORI SINGKAT

#### 2.1 Sistem Persamaan Linear (SPL)

##### 2.1.1 Metode Eliminasi Gauss

Metode Eliminasi Gauss adalah salah satu cara untuk menyelesaikan suatu SPL (sistem persamaan linear). Metode ini merupakan serangkaian OBE (Operasi Baris Elementer), yang berada dalam matriks. Penyelesaian matriks ini berupa bentuk eselon baris. Metode Eliminasi Gauss dapat dipakai untuk mendapatkan balikan matriks; selain itu, juga merupakan salah satu cara untuk menghitung determinan suatu matriks.

Ada tiga OBE yang dipakai:

- a. Pertukaran dua baris
- b. Perkalian/pembagian baris dengan angka/bilangan tertentu
- c. Penjumlahan/pengurangan baris dengan kelipatan dari baris lain.

Agar dapat memakai metode eliminasi ini, SPL dengan bentuk

$$\begin{aligned}
 a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n &= b_1 \\
 a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n &= b_2 \\
 a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \dots + a_{3n}x_n &= b_3 \\
 \vdots & \\
 a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nn}x_n &= b_n
 \end{aligned}$$

dapat dikonversi menjadi

$$\left[ \begin{array}{ccccc|c}
 a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\
 a_{21} & a_{22} & a_{23} & \dots & a_{2n} & b_2 \\
 a_{31} & a_{32} & a_{33} & \dots & a_{3n} & b_3 \\
 \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & b_n
 \end{array} \right]$$

Setelah itu, lakukan berbagai OBE hingga mendapat bentuk

$$\left[ \begin{array}{ccccc|c}
 1 & p_{12} & p_{13} & \dots & p_{1n} & q_1 \\
 0 & 1 & p_{23} & \dots & p_{2n} & q_2 \\
 0 & 0 & 1 & \dots & p_{3n} & q_3
 \end{array} \right]$$



$$\left[ \begin{array}{ccccc|c} \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & q_n \end{array} \right]$$

Dari matriks eselon tersebut, lakukan penyulihan mundur, yang berarti substitusi dari bawah ke atas. Nilai  $x_n = q_n$ ;  $x_{n-1} + p_{(n-1)(n)} x_n = q_{n-1}$ , sehingga  $x_{n-1} = q_{n-1} - p_{(n-1)(n)} x_n$ ; dan seterusnya sampai semua nilai variabel ditemukan (penyulihan sampai baris pertama).

Matriks tersebut dapat memiliki 0 solusi, 1 solusi, atau tak hingga solusi.

- Jika tiap baris memiliki *leading one*, maka matriks memiliki 1 solusi.
- Jika bagian kiri baris semuanya 0:
  - Jika bagian kanan baris 0, maka matriks memiliki 0 solusi.
  - Jika bagian kanan baris bukan 0, maka matriks memiliki tak hingga solusi.

### 2.1.2 Metode Eliminasi Gauss-Jordan

Metode Eliminasi Gauss-Jordan bisa didapatkan dengan mengikuti langkah-langkah Metode Eliminasi Gauss, dengan beberapa tambahan. Ubah bentuk matriks

$$\left[ \begin{array}{ccccc|c} 1 & p_{12} & p_{13} & \dots & p_{1n} & q_1 \\ 0 & 1 & p_{23} & \dots & p_{2n} & q_2 \\ 0 & 0 & 1 & \dots & p_{3n} & q_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & q_n \end{array} \right]$$

menjadi bentuk matriks

$$\left[ \begin{array}{ccccc|c} 1 & 0 & 0 & \dots & 0 & r_1 \\ 0 & 1 & 0 & \dots & 0 & r_2 \\ 0 & 0 & 1 & \dots & 0 & r_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & r_n \end{array} \right]$$

Dari bentuk tersebut, bisa didapatkan solusi  $x_1 = r_1$ ,  $x_2 = r_2$ , dan seterusnya.

### 2.1.3 Determinan

Determinan adalah suatu nilai yang hanya dimiliki oleh matriks persegi. Matriks yang jumlah baris dan kolomnya berbeda tidak memiliki determinan. Determinan sering ditulis dalam bentuk  $\det(A)$  atau  $|A|$ .

Ada dua cara untuk mendapat determinan, yaitu:

a. Reduksi baris

Pastikan matriks berbentuk diagonal atas terlebih dahulu (bisa didapatkan dengan metode eliminasi Gauss). Setelah itu, determinan bisa dicari dengan mengalikan semua nilai pada diagonal utama matriks dan seluruh k (perkalian suatu baris dengan bilangan k (OBE bagian b)).

Lalu hasil perkalian tersebut berbentuk positif jika jumlah pertukaran baris (c) genap atau tidak ada; negatif jika ganjil.

$$|A| = (-1)^c \times (k_1 \times k_2 \times k_3 \times \dots) \times (a_{11} \times a_{22} \times \dots \times a_{nn})$$

b. Ekspansi Kofaktor

Cari kofaktor (lihat V. Matriks Kofaktor) sepanjang satu baris (misalnya sepanjang baris 1:  $C_{11}, C_{12}, \dots, C_{1j}$ ) atau sepanjang satu kolom (misalnya sepanjang kolom 3:  $C_{13}, C_{23}, \dots, C_{i3}$ ). Lalu kalikan tiap  $C_{ij}$  dengan  $a_{ij}$ . Hasil perkalian tersebut ditambahkan.

$$|A| = \sum_{i=1}^n (a_{ij} \times C_{ij}) \text{ dengan } j \text{ suatu kolom yang dipilih.}$$

atau

$$|A| = \sum_{j=1}^n (a_{ij} \times C_{ij}) \text{ dengan } i \text{ suatu baris yang dipilih.}$$

### 2.1.4 Matriks Balikan

Matriks barisan atau matriks invers adalah matriks yang menghasilkan matriks identitas jika dikalikan dengan matriks awalnya. Matriks balikan sering ditulis dalam bentuk  $A^{-1}$ .

Ada dua cara untuk mendapat matriks balikan, yaitu:

a. Reduksi baris

Taruh matriks identitas di sebelah kanan matriks:

$$\left[ \begin{array}{ccccc|ccccc} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & 1 & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & 0 & 1 & 0 & \dots & 0 \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & 0 & 0 & 0 & \dots & 1 \end{array} \right]$$

Lalu, lakukan eliminasi Gauss-Jordan. Matriks sebelah kiri akan tereduksi, sedangkan matriks kanan akan menjadi matriks balikan.

b. Adjoin matriks

Adjoin matriks adalah transpose dari matriks kofaktor (lihat [V. Matriks Kofaktor](#)). Adjoin matriks sering ditulis dalam bentuk  $adj(A)$ .

$$adj(A) = C^T$$

Untuk mendapatkan matriks invers dengan metode adjoin matriks, adjoin matriks dibagi dengan determinan matriks.

$$A^{-1} = \frac{1}{\det(A)} \times adj(A)$$

### 2.1.5 Matriks Kofaktor

Matriks Kofaktor adalah positif atau negatif dari determinan submatriks. Kepositifan determinan submatriks tergantung oleh baris dan kolom. Jika baris dan kolom keduanya ganjil, atau keduanya genap, maka positif; jika keduanya berbeda (salah satu ganjil, salah satu genap), maka negatif.

$$C_{ij} = (-1)^{i+j} \times M_{ij}$$

Submatriks adalah matriks tanpa anggota dari baris i dan anggota dari kolom j.

## 2.2 Interpolasi Polinomial

Interpolasi Polinomial adalah cara mencari fungsi atau persamaan polinomial yang tepat sesuai dengan titik-titik koordinat pada data. Persamaan polinomial dapat ditulis sebagai berikut:

$$p(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

Misal kita diberikan data  $(x_1, y_1)$ ,  $(x_2, y_2)$ , dan  $(x_3, y_3)$ . Kita dapat mengubahnya menjadi

$$\left[ \begin{array}{ccc|c} x_1^0 & x_1 & x_1^2 & y_1 \\ x_2^0 & x_2 & x_2^2 & y_2 \\ x_3^0 & x_3 & x_3^2 & y_3 \end{array} \right]$$

Lalu, lakukan eliminasi Gauss-Jordan, sehingga  $a_0 = y_1$ ,  $a_1 = y_2$ , dan  $a_2 = y_3$ . Maka persamaan polinomialnya adalah  $p(x) = y_1 + y_2x + y_3x^2$  (dengan  $y$  adalah suatu bilangan, atau koefisien dari  $x$ ). Taksiran dari nilai  $x$  tertentu juga dapat dicari dengan cara mensubstitusikan nilai  $x$  tersebut ke persamaan.

## 2.3 Regresi Berganda

### 2.3.1 Regresi Linear Berganda

Regresi linear adalah salah satu cara yang dapat digunakan untuk memprediksi nilai, selain menggunakan interpolasi polinom. Rumus umum untuk mencari regresi linear berganda adalah

$$y_i = \beta_0 + \beta_1x_{1i} + \beta_2x_{2i} + \dots + \beta_kx_{ki} + \epsilon_i$$

Nilai setiap  $\beta$  dapat diperoleh dengan *normal estimation equation for multiple linear regression*, dilanjutkan dengan cara eliminasi Gauss.

Setiap  $n$ -peubah, jumlah variabel linier, kuadrat, dan interaksi akan berbeda-beda.

### 2.3.1 Regresi Kuadratik Berganda

Regresi Kuadratik Berganda adalah perluasan dari Regresi Linear Berganda yang memungkinkan variabel input yang berganda. Regresi ini juga memungkinkan hubungan non-linear antara variabel input dengan variabel targetnya. Bentuk persamaan regresi kuadratik ada tiga:

- Variabel Linier: Variabel dengan derajat satu (misalnya  $X, Y, Z$ )

- Variabel Kuadrat: Variabel dengan derajat dua (misalnya  $X^2$ )
- Variabel Interaksi: Perkalian 2 variabel dengan derajat satu (misalnya  $XY$ ,  $YZ$ ,  $XZ$ )

Contoh regresi kuadratik dengan dua variabel peubah:

$$\begin{pmatrix} N & \sum u_i & \sum v_i & \sum u_i^2 & \sum u_i v_i & \sum v_i^2 \\ \sum u_i & \sum u_i^2 & \sum u_i v_i & \sum u_i^3 & \sum u_i^2 v_i & \sum u_i v_i^2 \\ \sum v_i & \sum u_i v_i & \sum v_i^2 & \sum u_i^2 v_i & \sum u_i v_i^2 & \sum v_i^3 \\ \sum u_i^2 & \sum u_i^3 & \sum u_i^2 v_i & \sum u_i^4 & \sum u_i^3 v_i & \sum u_i^2 v_i^2 \\ \sum u_i v_i & \sum u_i^2 v_i & \sum u_i v_i^2 & \sum u_i^3 v_i & \sum u_i^2 v_i^2 & \sum u_i v_i^3 \\ \sum v_i^2 & \sum u_i v_i^2 & \sum v_i^3 & \sum u_i^2 v_i^2 & \sum u_i v_i^3 & \sum v_i^4 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum y_i u_i \\ \sum y_i v_i \\ \sum y_i u_i^2 \\ \sum y_i u_i v_i \\ \sum y_i v_i^2 \end{pmatrix}$$

$N$  adalah jumlah peubah.  $u_i$  dan  $v_i$  adalah variabel linear,  $u_i^2$  dan  $v_i^2$  adalah variabel kuadratik.

Regresi ini dapat diselesaikan dengan cara eliminasi Gauss.

## 2.4 Bicubic Spline Interpolation

Interpolasi Bikubik adalah metode interpolasi pada data dua dimensi yang biasanya dapat digunakan untuk perluasan data secara visual (misalnya *upscaling* gambar). Metode ini lebih akurat daripada Interpolasi Polinomial.

Misalnya, ada 16 buah titik: 4 titik utama  $f(0, 0)$ ,  $f(1, 0)$ ,  $f(0, 1)$ , dan  $f(1, 1)$ ; dan 12 titik aproksimasi di sekitarnya:  $f(-1, -1)$ ,  $f(0, -1)$ ,  $f(1, 0)$ , ..., dan  $f(2, 2)$ . Persamaan polinomialnya adalah

$$f(x, y) = \sum_{j=0}^3 \sum_{i=0}^3 (a_{ij} \times x^i \times y^j).$$

Dari fungsi tersebut, akan didapatkan persamaan

$$y = Xa$$

dengan matriks solusi  $16 \times 16$ .

# BAB 3

## IMPLEMENTASI PROGRAM

### 3.1 Sistem Persamaan Linear (SPL)

#### 3.1.1 Eliminasi Gauss

Fungsi ini mengembalikan solusi SPL yang direpresentasikan oleh matriks ini, menggunakan metode eliminasi Gauss.

##### 3.1.1.1 gaussEliminasi():

Fungsi ini mengimplementasikan metode eliminasi Gauss untuk menyelesaikan sistem persamaan linear yang diwakili oleh matriks augmented (koefisien dan konstanta). Tujuan utama dari fungsi ini adalah untuk mereduksi matriks menjadi bentuk eselon baris (row echelon form), kemudian menggunakan **back substitution** untuk menemukan solusi dari sistem persamaan.

##### 3.1.1.2 rowSwap (int i,int j):

Prosedur ini menukar dua baris matriks, digunakan saat pivot berada di baris yang berbeda dari posisi yang diinginkan.

##### 3.1.1.3 getElmt(int row, int col):

Fungsi ini mengembalikan elemen matriks pada indeks tertentu (row, col), digunakan untuk membaca nilai elemen pada matriks.

##### 3.1.1.4 setElmt(int row, int col, double value):

Prosedur ini mengubah elemen matriks pada indeks tertentu (row, col) dengan nilai baru (value), digunakan untuk memodifikasi matriks.

##### 3.1.1.5 banyakSolusi():

Prosedur ini menangani kasus di mana sistem memiliki banyak solusi atau solusi tak terbatas. Fungsinya adalah memberikan solusi yang berlaku dalam kondisi tersebut.

##### 3.1.1.6 backSubstitution(double[] X):

Fungsi ini melakukan back substitution untuk menyelesaikan sistem persamaan dari bentuk eselon atas. Nilai-nilai variabel disimpan dalam array X.

#### 3.1.1.7 `getRowLength()` dan `getColLength()`:

Fungsi ini digunakan untuk mendapatkan jumlah baris dan kolom dalam matriks, yang berguna untuk mengetahui dimensi matriks.

Langkah :

1. Pencarian Pivot:

Fungsi ini memulai dengan mencari elemen pivot (elemen pertama yang bukan nol) dalam setiap kolom. Jika ditemukan, elemen tersebut digunakan untuk mengeliminasi elemen-elemen di bawahnya pada kolom yang sama.

Jika tidak ada elemen non-nol di kolom tersebut, fungsi melanjutkan ke kolom berikutnya tanpa melakukan operasi lebih lanjut.

2. Penukaran Baris (Pivoting):

Jika pivot berada di baris yang berbeda dari posisi yang diinginkan, fungsi menukar baris pivot dengan baris saat ini menggunakan fungsi `rowSwap()`.

3. Normalisasi Pivot:

Setelah pivot ditemukan, baris pivot dibagi dengan nilai pivot untuk mengubah elemen pivot menjadi 1.

4. Eliminasi Baris di Bawah Pivot:

Untuk setiap elemen di bawah pivot, dilakukan eliminasi untuk membuat elemen-elemen di bawah pivot menjadi 0. Ini dilakukan dengan mengurangi baris pivot yang sudah dinormalisasi dari baris-baris di bawahnya.

5. Memindahkan Baris Nol ke Bawah:

Setelah melakukan eliminasi, baris yang seluruh elemennya nol dipindahkan ke bagian bawah matriks menggunakan `rowSwap()`.

6. Normalisasi Ulang Pivot:

Setelah proses eliminasi, baris pivot diperiksa lagi untuk memastikan bahwa pivot bernilai 1, dan dilakukan normalisasi ulang jika diperlukan.

7. Eliminasi Baris yang Sama atau Saling Menghapus:

Fungsi ini memeriksa apakah ada dua baris dengan nilai absolut elemen-elemen yang sama. Jika ditemukan, operasi penambahan atau

pengurangan dilakukan untuk mengeliminasi salah satu baris agar tidak menghasilkan sistem yang terduplikasi atau berlawanan tanda.

8. Menentukan Solusi:

Setelah matriks diubah menjadi bentuk eselon, sistem dapat diselesaikan. Jika jumlah baris lebih banyak dari jumlah kolom variabel (yang mengindikasikan adanya banyak solusi atau solusi tak terbatas), fungsi `banyakSolusi()` dipanggil untuk menangani kasus ini. Jika tidak, solusi unik dihitung menggunakan back substitution, dan hasilnya dicetak ke konsol.

### 3.1.2 Eliminasi Gauss-Jordan

Fungsi ini mengembalikan solusi SPL yang direpresentasikan oleh matriks ini, menggunakan metode eliminasi Gauss-Jordan.

#### 3.1.2.1 `gaussJordanEliminasi()`:

Mengimplementasikan metode eliminasi Gauss-Jordan, yang merupakan kelanjutan dari metode eliminasi Gauss. Tujuan dari metode ini adalah untuk mereduksi matriks menjadi bentuk eselon baris tereduksi (reduced row echelon form) atau yang juga dikenal dengan bentuk eselon baris yang tereduksi penuh, di mana semua elemen di atas dan di bawah pivot pada setiap kolom adalah nol, dan setiap pivot bernilai 1.

#### 3.1.2.2 `rowSwap(int i, int j)`:

Prosedur ini menukar dua baris matriks, digunakan ketika pivot berada di baris yang berbeda dari posisi saat ini dan perlu ditukar.

#### 3.1.2.3 `getElmt(int row, int col)`:

Fungsi ini mengembalikan elemen matriks pada indeks tertentu (row, col), digunakan untuk membaca nilai elemen matriks pada proses eliminasi dan normalisasi.

#### 3.1.2.4 `setElmt(int row, int col, double value)`:

Prosedur ini mengubah elemen matriks pada indeks tertentu (row, col) dengan nilai baru (value), digunakan untuk memodifikasi elemen matriks setelah dilakukan operasi pembagian atau eliminasi.

#### 3.1.2.5 `getRowLength()` dan `getColLength()`:

Fungsi ini mengembalikan jumlah baris dan kolom dalam matriks, yang penting untuk menentukan dimensi matriks saat melakukan operasi.



Langkah :

1. Pencarian Pivot: Setiap baris mencari pivot di kolom yang sesuai. Jika ditemukan, pivot dipindahkan ke posisi yang benar.
2. Penukaran Baris: Pivot mungkin harus dipindahkan melalui penukaran baris jika berada di luar posisi yang diinginkan.
3. Normalisasi Pivot: Pivot diubah menjadi 1 dengan membagi seluruh baris dengan nilai pivot.
4. Eliminasi Atas dan Bawah Pivot: Elemen-elemen di baris lain dalam kolom yang sama diatur menjadi nol melalui pengurangan baris dengan kelipatan baris pivot.
5. Memindahkan Baris Nol: Setelah eliminasi, baris dengan semua elemen nol dipindahkan ke bawah matriks.
6. Normalisasi Akhir: Setiap baris dengan pivot yang masih tidak bernilai 1 di-normalisasi ulang untuk memastikan semua pivot bernilai 1.

### 3.1.3 Cramer

Menerima input sebuah matriks yang berisi koefisien dan matriks yang berisi konstanta secara terpisah atau matriks augmented yang berisi koefisien dan konstanta secara bersamaan. Kemudian, fungsi menghitung solusi system persamaan linear dengan menggunakan determinan matriks. Nilai yang dikembalikan fungsi berupa tipe bentukan SolusiSPL yang berupa list yang berisi solusi persamaan linear.

#### 3.1.3.1 cramer():

Menerapkan Aturan Cramer, yang merupakan metode untuk menyelesaikan sistem persamaan linear. Metode ini menggunakan determinan dari matriks untuk menghitung nilai variabel dalam sistem persamaan.

#### 3.1.3.2 matrixA():

Fungsi ini mengembalikan matriks koefisien AAA dari sistem persamaan yang ingin diselesaikan. Ini adalah bagian dari objek saat ini (biasanya matriks augmented).

#### 3.1.3.3 matrixb():

Fungsi ini mengembalikan vektor kolom bbb dari sistem persamaan, yang berisi nilai-nilai konstanta di sisi kanan dari persamaan.

#### 3.1.3.4 copyMat():

Metode ini digunakan untuk membuat salinan dari matriks A sehingga perubahan pada  $A_i$  tidak mempengaruhi A aslinya.

#### 3.1.3.5 determinantCof():

Fungsi ini menghitung determinan dari matriks menggunakan metode kofaktor, yang diperlukan untuk menyelesaikan sistem persamaan. Akan dijelaskan lebih lanjut di laporan.

Langkah :

1. Ambil Matriks Koefisien dan Vektor Konstanta: Matriks A dan vektor b diambil dari sistem persamaan.
2. Hitung Determinan Matriks A: Menggunakan metode determinantCof().
3. Buat Matriks  $A_i$ : Untuk setiap kolom dalam A:
  - a. Salin A ke  $A_i$ .
  - b. Ganti kolom j di  $A_i$  dengan vektor b.
  - c. Hitung determinan dari  $A_i$ .
4. Hitung Solusi untuk Setiap Variabel: Solusi untuk setiap variabel adalah rasio determinan  $A_i$  dengan determinan A.
5. Kembalikan Hasil Solusi: Hasil akhir adalah matriks  $\text{detXi}$ , yang berisi nilai solusi untuk setiap variabel dalam sistem persamaan linear.

## 3.2. Determinan

### 3.2.1 Determinan Kofaktor

#### 3.2.1.1 determinantCof()

menghitung determinan dari matriks menggunakan metode kofaktor. Metode ini merupakan pendekatan rekursif untuk menemukan determinan matriks persegi dengan menghitung determinan dari submatriks yang dihasilkan dengan menghapus baris dan kolom tertentu.

#### 3.2.1.2 retrieveELMT(int row, int col)

Fungsi ini digunakan untuk mendapatkan elemen pada posisi (row, col) dari matriks saat ini. Ini diperlukan untuk mengakses elemen dari matriks yang sedang diproses.

#### 3.2.1.3 getCofactor(Matrix subMat, int row, int col, int n):

Fungsi ini bertanggung jawab untuk mengisi matriks subMat dengan kofaktor dari elemen pada posisi (row, col) di matriks asli. Kofaktor dihitung dengan menghapus baris dan kolom yang terkait dengan elemen tersebut.

Langkah :

1. Pengecekan Matriks Persegi:  
Fungsi pertama-tama memeriksa apakah matriks adalah persegi dengan memeriksa apakah jumlah baris (nRow) sama dengan jumlah kolom (nCol). Jika tidak, fungsi akan mengembalikan 0, menunjukkan bahwa determinan tidak dapat dihitung.
2. Kasus Dasar:  
Jika matriks memiliki ukuran 1x1 (hanya satu elemen), determinan sama dengan elemen tersebut, dan fungsi akan mengembalikannya.
3. Inisialisasi Variabel:  
Variabel det diinisialisasi untuk menyimpan nilai determinan yang dihitung. Variabel plusMinus diinisialisasi untuk mengelola tanda dari setiap elemen yang terlibat dalam perhitungan determinan.
4. Perulangan Melalui Kolom: Fungsi melakukan perulangan melalui setiap kolom dari baris pertama (baris indeks 0):  
Jika elemen pada posisi (0, j) tidak sama dengan nol, langkah-langkah berikut diambil:
  - Sebuah submatriks subMat dibuat, yang memiliki ukuran satu baris dan satu kolom lebih kecil dari matriks asli.
  - Fungsi getCofactor() dipanggil untuk mengisi subMat dengan kofaktor dari elemen (0, j).
  - Determinan dari subMat dihitung secara rekursif dengan memanggil determinantCof().
  - Nilai determinan ditambahkan ke det dengan memperhitungkan tanda (plusMinus) dan elemen (0, j).
5. Mengubah Tanda:  
Setelah setiap iterasi, nilai plusMinus dikalikan dengan -1 untuk memastikan tanda bergantian (positif-negatif).
6. Keluaran:  
Setelah semua kolom diproses, fungsi mengembalikan nilai det, yang merupakan determinan dari matriks.

### 3.2.2 Determinan Reduksi Baris

#### 3.2.2.1 determinantRed()

Menghitung determinan dari matriks dengan menggunakan metode reduksi baris (row reduction). Metode ini mengubah matriks menjadi bentuk segitiga (atau bentuk echelon) dan kemudian menghitung determinan berdasarkan elemen diagonal.

#### 2.2.2 copyMat():

Fungsi ini digunakan untuk membuat salinan dari matriks, sehingga perubahan yang dilakukan pada matriks salinan tidak mempengaruhi matriks asli.

### 2.2.3 rowRed():

Fungsi ini bertanggung jawab untuk melakukan reduksi baris pada matriks. Ini mengubah matriks menjadi bentuk segitiga atau echelon, yang diperlukan untuk perhitungan determinan.

Langkah :

1. Inisialisasi Variabel:

Variabel swapCount direset ke 0, yang digunakan untuk melacak jumlah pertukaran baris yang terjadi selama proses reduksi. Ini penting untuk menentukan tanda dari determinan. Variabel divMult direset ke 1.0, yang digunakan untuk mengalikan faktor-faktor yang terlibat dalam pembagian saat mengubah elemen menjadi 1 selama proses reduksi.

2. Salinan Matriks:

Sebuah salinan dari matriks asli dibuat menggunakan copyMat() agar matriks asli tidak berubah selama proses perhitungan.

3. Reduksi Baris:

Fungsi rowRed() dipanggil pada salinan matriks (mReduced). Fungsi ini melakukan reduksi baris untuk mengubah matriks menjadi bentuk segitiga. Ini termasuk membuat elemen diagonal utama menjadi 1 (normalisasi) dan menghilangkan elemen di bawahnya.

4. Perhitungan Determinan:

Variabel detMult diinisialisasi ke 1.0 untuk menyimpan hasil perkalian elemen diagonal dari matriks yang sudah direduksi.

Fungsi kemudian melakukan perulangan melalui setiap baris dari matriks mReduced, mengalikan elemen diagonal utama (yaitu elemen (i, i)) ke dalam detMult.

5. Tanda Determinan:

Tanda dari determinan dihitung berdasarkan jumlah pertukaran baris (swapCount). Jika jumlah pertukaran adalah ganjil, plusMinus diatur ke -1 (negatif), jika genap, diatur ke 1 (positif).

6. Menghitung Nilai Akhir Determinan:

Determinan akhir dihitung dengan mengalikan plusMinus, divMult, dan detMult. Ini memberikan nilai determinan dari matriks yang telah direduksi.

7. Keluaran:

Fungsi ini mengembalikan nilai determinan yang dihitung.

## 3. 3 Matriks Balikan

### 3.3.1 Matriks Balikan OBE

### 3.3.1.2 inversOBE():

Menghitung invers dari sebuah matriks menggunakan metode Operasi Baris Elementer (Elementary Row Operations, OBE). Fungsi ini mengubah matriks input menjadi matriks identitas sambil menerapkan operasi yang sama pada matriks identitas, sehingga menghasilkan invers matriks.

### 3.3.1.2 getRowLength() dan getColLength():

Fungsi ini digunakan untuk mendapatkan jumlah baris dan kolom dari matriks, yang penting untuk menentukan ukuran matriks invers.

### 3.3.1.3 getElmt(i, j) dan setElmt(i, j, value):

Fungsi ini digunakan untuk mengambil dan mengatur elemen pada posisi tertentu dalam matriks. Ini memungkinkan akses langsung ke elemen matriks.

### 3.3.1.4 rowSwap(i, k):

Fungsi ini digunakan untuk menukar dua baris dalam matriks. Ini diperlukan ketika elemen pivot adalah nol, untuk memastikan bahwa proses eliminasi dapat dilanjutkan.

Langkah :

1. Inisialisasi Matriks Invers:

Matriks baru (`mat_inv`) dibuat dengan ukuran yang sama dengan matriks input tetapi dengan dua kali jumlah kolom (untuk menyimpan matriks input dan matriks identitas). Variabel `i`, `j`, dan `k` digunakan sebagai indeks dalam loop.

2. Mengisi Matriks:

Loop pertama digunakan untuk mengisi `mat_inv`. Kolom pertama hingga kolom tengah diisi dengan elemen dari matriks input, sedangkan kolom kedua hingga akhir diisi dengan elemen matriks identitas (1 pada diagonal utama dan 0 di tempat lain).

3. Proses Eliminasi:

Loop kedua bertujuan untuk mendapatkan invers dari matriks. Dalam loop ini, untuk setiap baris `i`, jika ada baris `j` yang bukan baris pivot (artinya `i != j`), proses dilanjutkan: Jika elemen pivot (`mat_inv[i][i]`) adalah 0, dilakukan pencarian baris di bawahnya (`k`) untuk menemukan baris yang memiliki elemen tidak nol di kolom `i`, kemudian menukar baris tersebut dengan baris `i`. Selanjutnya, elemen di baris `j` diubah untuk menghilangkan elemen di kolom `i`. Ini dilakukan dengan menghitung nilai pengali (sub) dan menerapkan operasi untuk mengurangi baris `i` dari baris `j`.

4. Normalisasi Baris Pivot:  
Setelah menyelesaikan eliminasi, loop ketiga digunakan untuk menormalkan setiap baris. Setiap elemen di baris  $i$  dibagi dengan elemen pivot (buatbagi) untuk memastikan bahwa elemen pivot menjadi 1.
5. Mengisi Matriks Asli dengan Invers:  
Loop terakhir menyalin hasil invers (dari kolom identitas yang telah diproses) kembali ke matriks asli. Matriks asli sekarang berisi invers dari matriks input.
6. Keluaran:  
Fungsi ini mengembalikan referensi ke matriks asli yang sekarang berisi invers matriks.

### 3.3.2 Matriks Balikan Adjoin

#### 3.3.2.1 inversadj():

menghitung invers dari sebuah matriks menggunakan metode Matriks Adjungat (Adjugate Matrix) dan determinan. Jika determinan dari matriks tersebut adalah nol, fungsi ini akan mengembalikan nilai null, menandakan bahwa invers tidak ada. Berikut adalah penjelasan rinci mengenai proses dan fungsi-fungsi yang digunakan dalam kode ini.

#### 3.3.2.2 determinantCof():

Fungsi ini digunakan untuk menghitung determinan dari matriks input. Hasilnya digunakan untuk memeriksa apakah matriks memiliki invers atau tidak.

#### 3.3.2.3 adjugate():

Fungsi ini digunakan untuk menghitung matriks adjungat dari matriks input. Matriks ini penting untuk menghitung invers, karena invers dapat dinyatakan sebagai hasil bagi antara matriks adjungat dan determinan.

#### 3.3.2.4 setElmt(i, j, value) dan getElmt(i, j):

Fungsi ini digunakan untuk mengakses dan mengatur elemen pada posisi tertentu dalam matriks, memungkinkan operasi langsung pada elemen matriks.

Langkah :

1. Hitung Determinan:

Pertama, fungsi ini memanggil metode `determinantCof()` untuk menghitung determinan dari matriks. Jika determinan sama dengan nol ( $\text{det} == 0$ ), ini menunjukkan bahwa matriks tidak memiliki invers (matriks singular). Jika determinan adalah nol, fungsi mencetak pesan dan mengembalikan null.

2. Inisialisasi Matriks Invers:

Jika determinan tidak nol, fungsi akan membuat matriks baru `mat_inv` dengan ukuran yang sama dengan matriks input (`nRow x nCol`).

3. Menghitung Adjugate :

Fungsi memanggil metode `adjugate()` untuk mendapatkan matriks adjungat dari matriks input. Matriks adjungat adalah matriks yang terdiri dari kofaktor dari setiap elemen, di mana posisi elemen dipertukarkan.

4. Normalisasi dengan Determinan:

Dengan menggunakan loop ganda, elemen-elemen dari matriks adjungat dibagi dengan determinan untuk menghasilkan elemen invers. Setiap elemen `mat_inv[i][j]` diisi dengan hasil pembagian elemen dari adjugate (`adj.getElmt(i, j)`) dengan determinan (`det`).

5. Keluaran:

Fungsi ini mengembalikan matriks invers yang telah dihitung (`mat_inv`)

### 3.4. Interpolasi Polinom

#### 3.4.1 Interpolasi Polinom

Mengelola data berupa titik-titik yang akan digunakan dalam interpolasi. Interpolasi polinomial digunakan untuk membangun polinomial yang sesuai dengan kumpulan titik yang diberikan.

##### 3.4.1.1 minX():

Mengembalikan nilai minimum dari kolom x dari matriks yang diberikan.

##### 3.4.1.2 maxX():

Mengembalikan nilai maksimum dari kolom x dari matriks yang diberikan.

##### 3.4.1.3 allPointsDifferent():

Memeriksa apakah semua titik yang diberikan adalah unik. Jika ada dua titik yang sama, maka akan dikembalikan false. Tujuannya untuk memastikan interpolasi polinomial berjalan dengan benar.

#### 3.4.1.4 polynomialInterpolation():

menjalankan interpolasi polinomial. Terdapat dua opsi yang diberikan kepada pengguna. Opsi 1: Pengguna memasukkan titik secara manual dan opsi 2: Pengguna membaca titik dari file.

Setelah membaca titik, matriks interpolasi dibentuk dengan memasukkan nilai  $x$  dari tiap titik ke dalam matriks polinomial. Matriks ini kemudian diselesaikan menggunakan metode eliminasi Gauss untuk mendapatkan koefisien polinomial. Dihitung nilai  $y$  untuk nilai  $x$  yang dimasukkan pengguna (baik dari file atau manual). Jika  $x$  berada di luar batas  $x_0$  hingga  $x_n$  atau jika titik-titik tidak unik, proses dihentikan. Hasil akhirnya adalah polinomial interpolasi yang terbentuk (misalnya  $f(x) = 2.5x^2 + 1.5x + 0.5$ ) serta nilai fungsi  $f(x)$  pada titik  $x$  yang diminta.

### 3.5. Interpolasi Bicubic Spline

#### 3.5.1 Interpolasi Bicubic

##### 3.5.1.1 xMat() :

Membuat matriks 16x16 dengan elemen yang dihitung menggunakan nilai  $x$ ,  $y$ ,  $k$ , dan  $l$ .

##### 3.5.1.1.1 Matrix(int row, int col):

Constructor untuk membuat objek matriks dengan ukuran baris dan kolom yang ditentukan (16x16 dalam hal ini).

##### 3.5.1.1.2 getRowLength():

Mengembalikan panjang (jumlah baris) dari matriks.

##### 3.5.1.1.3 getColLength():

Mengembalikan panjang (jumlah kolom) dari matriks.

##### 3.5.1.1.4 setElmt(int row, int col, double value):

Mengatur nilai dari elemen matriks pada posisi baris  $row$  dan kolom  $col$  dengan nilai  $value$ .

Langkah :

1. Inisialisasi matriks 16x16.



2. Looping baris i dan kolom j untuk mengisi elemen-elemen.
3. Nilai elemen dihitung tergantung pada rentang baris i(0-3, 4-7, 8-11, 12-15) dengan menggunakan aturan perhitungan yang berbeda (melibatkan x, y, k, l).
4. Looping pada eksponen k dan l digunakan untuk memvariasikan nilai eksponen dalam perhitungan elemen.

#### 3.5.1.2 array():

Mengubah matriks 16x16 menjadi array (matriks 16x1) dengan menyusun elemen-elemen matriks secara linear dari baris pertama hingga terakhir.

##### 5.1.2.1 Matrix(int row, int col):

Constructor untuk membuat objek matriks dengan ukuran baris dan kolom yang ditentukan (16x16 dalam hal ini).

##### 3.5.1.2.2 getRowLength():

Mengembalikan panjang (jumlah baris) dari matriks.

##### 3.5.1.2.3 getColLength():

Mengembalikan panjang (jumlah kolom) dari matriks.

##### 3.5.1.2.4 setElmt(int row, int col, double value):

Menyimpan nilai elemen pada matriks array baru (16x1) di posisi yang ditentukan.

##### 3.5.1.2.5 getElmt(int row, int col):

Mengambil nilai dari elemen matriks pada posisi baris Mengambil nilai dari elemen matriks pada posisi baris row dan kolom col.

Langkah :

1. Buat array baru arr berukuran 16x1.
2. Looping pada matriks input mat (16x16) secara linear.
3. Setiap elemen matriks input dipindahkan ke matriks arr secara urut.

#### 3.5.1.3 result():

menghitung hasil berdasarkan matriks 16x1 dan dua parameter mmm dan nnn. Setiap elemen matriks dikalikan dengan  $m^i \cdot n^j$  (dengan i dan j sebagai eksponen), lalu semua hasil perkalian dijumlahkan untuk mendapatkan nilai akhir. Fungsi ini

dirancang untuk memproses nilai-nilai dari matriks dan memberikan hasil akhir berupa perhitungan matematis kompleks yang melibatkan dua variabel.

3.5.1.3.1 `getElmt(int row, int col):`

Mengambil nilai elemen dari matriks 16x1 pada baris yang ditentukan.

3.5.1.2.2 `Math.pow(double a, double b):`

Fungsi untuk menghitung pangkat  $a^b$  (digunakan untuk  $m^i$  dan  $n^j$ ).

3.5.1.2.3 `getColLength():`

Mengembalikan panjang (jumlah kolom) dari matriks.

3.5.1.2.4 `setElmt(int row, int col, double value):`

Mengatur nilai dari elemen matriks pada posisi baris row dan kolom col dengan nilai value.

3.5.1.2.5 `Math.pow(double a, double b):`

Fungsi bawaan Java yang mengembalikan nilai pangkat  $a^b$  (digunakan untuk operasi eksponen).

Langkah :

1. Looping untuk eksponen j (0 hingga 3) untuk variabel nnn.
2. Di dalamnya, looping untuk eksponen i (0 hingga 3) untuk variabel mmm.
3. Pada setiap iterasi, ambil elemen dari matriks 16x1 dan kalikan dengan  $m^i \cdot n^j$
4. Hasil setiap perkalian ditambahkan ke variabel result.
5. Setelah semua iterasi selesai, kembalikan nilai akhir result.

## **3.6. Regresi Linear dan Kudratik**

### **3.6.1 Regresi Linear**

Mengimplementasikan regresi linear berganda menggunakan matriks untuk menyelesaikan persamaan normal dan menghitung hasil regresi. Terdapat beberapa metode yang bekerja bersama-sama untuk melakukan langkah-langkah tersebut, mulai dari menyusun matriks persamaan normal hingga menghitung hasil regresi pada nilai yang diberikan.

3.6.1.1 `MultiLinearReg(Matrix mIn):`

Fungsi ini bertugas membentuk **matriks persamaan normal** dari regresi linear berganda. Matriks persamaan normal ini diperlukan untuk menghitung koefisien regresi. Setiap elemen dari matriks ini adalah hasil dari operasi penjumlahan dan perkalian antara elemen-elemen dari matriks input.

#### 3.6.1.1.1 getColLength() dan getRowLength():

Digunakan untuk mendapatkan jumlah baris dan kolom dari matriks input mIn.

#### 3.6.1.1.2 getElmt(i, j)

Digunakan untuk mendapatkan nilai dari elemen di posisi (i,j)(i, j)(i,j) dalam matriks mIn.

#### 3.6.1.1.3 setElmt(k, j, value)

Digunakan untuk menetapkan nilai hasil perhitungan ke elemen pada posisi (k,j)(k, j)(k,j) dalam matriks m\_normalEstimation.

### 3.6.1.2 MsolveReg(Matrix mat):

Menyelesaikan sistem persamaan normal untuk **menghitung koefisien regresi**.

Koefisien ini nantinya akan digunakan untuk membuat persamaan regresi yang dapat memprediksi variabel dependen.

#### 3.6.1.2.1 MultiLinearReg(Matrix mIn)

Digunakan untuk mendapatkan jumlah baris dan kolom dari matriks input mIn.

#### 3.6.1.2.2 inverseGab():

Digunakan untuk menghitung invers dari matriks persamaan normal, yang merupakan solusi dari sistem persamaan linear.

### 3.6.1.3 getRegEq(Matrix mat):

membentuk persamaan regresi dalam bentuk string, berdasarkan koefisien yang dihitung sebelumnya. Persamaan regresi ini menggambarkan hubungan antara variabel independen dan variabel dependen.

#### 3.6.1.3.1 getColLength() dan getRowLength():

Digunakan untuk mendapatkan dimensi dari matriks koefisien.

#### 3.6.1.3.2 getElmt(i, j)

Digunakan untuk mendapatkan nilai dari elemen pada posisi  $(i,j)(i, j)(i,j)$  dalam matriks koefisien.

#### 3.6.1.3.3 setElmt(k, j, value)

Digunakan untuk menetapkan nilai hasil perhitungan ke elemen pada posisi  $(k,j)(k, j)(k,j)$  dalam matriks `m_normalEstimation`.

#### 3.6.1.4 TaksiranReg(Matrix mat, double[] val):

menghitung taksiran hasil regresi pada titik-titik tertentu yang diberikan sebagai input. Hasil ini merepresentasikan prediksi dari variabel dependen berdasarkan nilai-nilai variabel independen.

##### 3.6.1.4.1 getRowLength():

Digunakan untuk mendapatkan jumlah baris dari matriks koefisien.

##### 3.6.1.4.2 getElmt(i, j):

Digunakan untuk mendapatkan nilai elemen dari matriks koefisien pada posisi  $(i,j)(i, j)(i,j)$ .

#### 3.6.1.5 readMtrx(Matrix mat)

membaca input dari pengguna untuk mengisi matriks dengan data yang nantinya digunakan dalam perhitungan regresi.

##### 3.6.1.5.1 getRowLength() dan getColLength()

Digunakan untuk mengetahui dimensi matriks `mat`.

##### 3.6.1.5.2 setElmt(i, j, value):

Digunakan untuk menetapkan nilai input ke posisi  $(i,j)(i, j)(i,j)$  dalam matriks.

##### 3.6.1.5.3 Scanner sc = new Scanner(System.in):

Digunakan untuk membaca input dari pengguna.

### 3.6.2 Regresi Kuadratik



## BAB 4

### EKSPERIMEN

#### 4.1. SPL $Ax = B$

##### 4.1.1.Studi kasus 1

$$A = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 5 & -7 & -5 \\ 2 & -1 & 1 & 3 \\ 5 & 2 & -4 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -2 \\ 4 \\ 6 \end{bmatrix}$$

Jawaban :

Solusi tunggal:

$$X[1] = 1.7333$$

$$X[2] = -0.4333$$

$$X[3] = 0.4000$$

$$X[4] = -0.1000$$

Matriks :

$$1.0 \ 1.0 \ -1.0 \ -1.0 \ 1.0$$

$$0.0 \ 1.0 \ -1.6666666666666667 \ 2.3333333333333335 \ -1.3333333333333333$$

$$0.0 \ 0.0 \ 1.0 \ -6.0 \ 1.0$$

$$0.0 \ 0.0 \ 0.0 \ 1.0 \ -0.1$$

##### 4.1.2.Studi kasus 2

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & -3 & 0 \\ 2 & -1 & 0 & 1 & -1 \\ -1 & 2 & 0 & -2 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 6 \\ 5 \\ -1 \end{bmatrix}$$

Jawaban :

##### 4.1.3.Studi kasus 3

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$

**Jawaban :**

#### 4.1.3. Studi kasus 4

$$H = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \dots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \dots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \dots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \dots & \frac{1}{2n+1} \end{bmatrix} \quad \underline{=} \quad b = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

H adalah matriks *Hilbert*. Cobakan untuk  $n = 6$  dan  $n = 10$ .

**Jawaban :**

## 4.2 SPL Augmented

### 4.2.1. Studi kasus 5

$$\begin{bmatrix} 1 & -1 & 2 & -1 & -1 \\ 2 & 1 & -2 & -2 & -2 \\ -1 & 2 & -4 & 1 & 1 \\ 3 & 0 & 0 & -3 & -3 \end{bmatrix}.$$

**Jawaban :**

### 4.2.2. Studi kasus 6

$$\begin{bmatrix} 2 & 0 & 8 & 0 & 8 \\ 0 & 1 & 0 & 4 & 6 \\ -4 & 0 & 6 & 0 & 6 \\ 0 & -2 & 0 & 3 & -1 \\ 2 & 0 & -4 & 0 & -4 \\ 0 & 1 & 0 & -2 & 0 \end{bmatrix}.$$

**Jawaban :**

## 4.3 SPL Berbentuk

### 4.3.1 Studi kasus 7

$$8x_1 + x_2 + 3x_3 + 2x_4 = 0$$

$$2x_1 + 9x_2 - x_3 - 2x_4 = 1$$

$$x_1 + 3x_2 + 2x_3 - x_4 = 2$$

$$x_1 + 6x_3 + 4x_4 = 3$$

Jawaban :

```
Masukkan jumlah baris: 4
Masukkan jumlah kolom: 5
8 1 3 2 0
2 9 -1 -2 1
1 3 2 -1 2
1 0 6 4 3
Solusi:
X1 = -0.2243
X2 = 0.1824
X3 = 0.7095
X4 = -0.2581
Matriks :
1.0 0.125 0.375 0.25 0.0
0.0 1.0 -0.2 -0.2857142857142857 0.11428571428571428
0.0 0.0 1.0 -0.19480519480519481 0.7597402597402597
0.0 0.0 0.0 1.0 -0.25810810810810797
```

### 4.3.2 Studi kasus 8

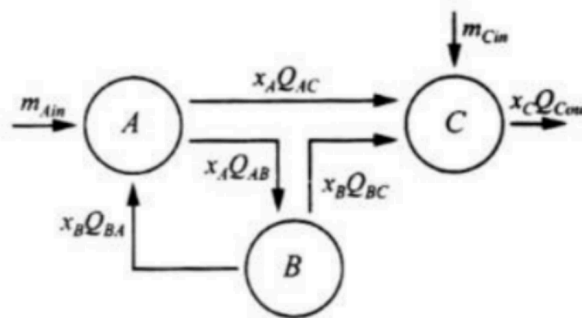


$$\begin{aligned}
x_7 + x_8 + x_9 &= 13.00 \\
x_4 + x_5 + x_6 &= 15.00 \\
x_1 + x_2 + x_3 &= 8.00 \\
0.04289(x_3 + x_5 + x_7) + 0.75(x_6 + x_8) + 0.61396x_9 &= 14.79 \\
0.91421(x_3 + x_5 + x_7) + 0.25(x_2 + x_4 + x_6 + x_8) &= 14.31 \\
0.04289(x_3 + x_5 + x_7) + 0.75(x_2 + x_4) + 0.61396x_1 &= 3.81 \\
x_3 + x_6 + x_9 &= 18.00 \\
x_2 + x_5 + x_8 &= 12.00 \\
x_1 + x_4 + x_7 &= 6.00 \\
0.04289(x_1 + x_5 + x_9) + 0.75(x_2 + x_6) + 0.61396x_3 &= 10.51 \\
0.91421(x_1 + x_5 + x_9) + 0.25(x_2 + x_4 + x_6 + x_8) &= 16.13 \\
0.04289(x_1 + x_5 + x_9) + 0.75(x_4 + x_8) + 0.61396x_7 &= 7.04
\end{aligned}$$

Jawaban :

## 4.4 Sistem reaktor

### 4.4.1 Studi kasus 9



Dengan laju volume  $Q$  dalam  $m^3/s$  dan input massa min dalam  $mg/s$ . Konservasi massa pada tiap inti reaktor adalah sebagai berikut:

$$A: m_{Ain} + Q_{BA}x_B - Q_{AB}x_A - Q_{AC}x_A = 0$$

$$B: Q_{AB}x_A - Q_{BA}x_B - Q_{BC}x_B = 0$$

$$C: m_{Cin} + Q_{AC}x_A + Q_{BC}x_B - Q_{Cout}x_C = 0$$

Tentukan solusi  $x_A$ ,  $x_B$ ,  $x_C$  dengan menggunakan parameter berikut :  $Q_{AB} = 40$ ,  $Q_{AC} = 80$ ,  $Q_{BA} = 60$ ,  $Q_{BC} = 20$  dan  $Q_{Cout} = 150 m^3/s$  dan  $m_{Ain} = 1300$  dan  $m_{Cin} = 200 mg/s$ .

**Jawaban :**

$$1300 + 60x_B - 40x_A - 80x_A = 0 \quad \rightarrow \quad -120x_A + 60x_B = -1300$$

$$40x_A - 60x_B - 20x_B = 0 \quad \rightarrow \quad 40x_A - 80x_B = 0$$

$$200 + 80x_A + 20x_B - 150x_{Cout} = 0 \quad \rightarrow \quad 80x_A + 20x_B - 150x_{Cout} = -200$$

```
Masukkan jumlah baris: 3
Masukkan jumlah kolom: 4
-120 60 0 -1300
40 -80 0 0
80 20 -150 -200
Solusi:
X1 = 14.4444
X2 = 7.2222
X3 = 10.0000
Matriks :
1.0 -0.5 -0.0 10.833333333333334
0.0 1.0 -0.0 7.222222222222223
0.0 0.0 1.0 10.0
```

## 4.5 Interpolasi Polinom

### 4.5.1. Studi Kasus 10:

Gunakan tabel di bawah ini untuk mencari polinom interpolasi dari pasangan titik-titik yang terdapat dalam tabel. Program menerima masukan nilai  $x$  yang akan dicari nilai fungsi  $f(x)$ .

$x$	0.1	0.3	0.5	0.7	0.9	1.1	1.3
$f(x)$	0.003	0.067	0.148	0.248	0.370	0.518	0.697

Lakukan pengujian pada nilai-nilai berikut:

$$x = 0.2 \quad f(x) = ?$$

$$x = 0.55 \quad f(x) = ?$$

$$x = 0.85 \quad f(x) = ?$$

$$x = 1.28 \quad f(x) = ?$$

**Jawaban :**

```
Masukkan banyak titik: 7
0.1 0.003
0.3 0.067
0.5 0.148
0.7 0.248
0.9 0.370
1.1 0.518
1.3 0.697
0.2
f(x) = (-1.9674693069285452E-14)x^6 + (8.37461474950599E-14)x^5 + (0.0260416666652843)x^4 + (1.1103618025032347E-13)x^3
+ (0.1973958333328886)x^2 + (0.24000000000000804)x^1 + -0.02297656250000046, f(0.2) = 0.032960937500000065
```

```
Masukkan banyak titik: 7
0.1 0.003
0.3 0.067
0.5 0.148
0.7 0.248
0.9 0.370
1.1 0.518
1.3 0.697
0.55
f(x) = (-1.9674693069285452E-14)x^6 + (8.37461474950599E-14)x^5 + (0.0260416666652843)x^4 + (1.1103618025032347E-13)x^3
+ (0.1973958333328886)x^2 + (0.24000000000000804)x^1 + -0.02297656250000046, f(0.55) = 0.17111865234375
```

```
Masukkan banyak titik: 7
0.1 0.003
0.3 0.067
0.5 0.148
0.7 0.248
0.9 0.370
1.1 0.518
1.3 0.697
0.85
f(x) = (-1.9674693069285452E-14)x^6 + (8.37461474950599E-14)x^5 + (0.0260416666652843)x^4 + (1.1103618025032347E-13)x^3
+ (0.1973958333328886)x^2 + (0.24000000000000804)x^1 + -0.02297656250000046, f(0.85) = 0.33723583984375005
```

```
Masukkan banyak titik: 7
0.1 0.003
0.3 0.067
0.5 0.148
0.7 0.248
0.9 0.370
1.1 0.518
1.3 0.697
1.28
f(x) = (-1.9674693069285452E-14)x^6 + (8.37461474950599E-14)x^5 + (0.0260416666652843)x^4 + (1.1103618025032347E-13)x^3
+ (0.1973958333328886)x^2 + (0.24000000000000804)x^1 + -0.02297656250000046, f(1.28) = 0.6775418375
```

#### 4.5.2 Studi Kasus 11:

- a. Jumlah kasus positif baru Covid-19 di Indonesia semakin fluktuatif dari hari ke hari. Di bawah ini diperlihatkan jumlah kasus baru Covid-19 di Indonesia mulai dari tanggal 17 Juni 2022 hingga 31 Agustus 2022:

Tanggal	Tanggal (desimal)	Jumlah Kasus Baru
17/06/2022	6,567	12.624
30/06/2022	7	21.807

08/07/2022	7,258	38.391
14/07/2022	7,451	54.517
17/07/2022	7,548	51.952
26/07/2022	7,839	28.228
05/08/2022	8,161	35.764
15/08/2022	8,484	20.813
22/08/2022	8,709	12.408
31/08/2022	9	10.534

Tanggal (desimal) adalah tanggal yang sudah diolah ke dalam bentuk desimal 3 angka di belakang koma dengan memanfaatkan perhitungan sebagai berikut:

$$\text{Tanggal (desimal)} = \text{bulan} + (\text{tanggal} / \text{jumlah hari pada bulan tersebut})$$

Sebagai contoh, untuk tanggal 17/06/2022 (dibaca: 17 Juni 2022) diperoleh tanggal(desimal) sebagai berikut:

$$\text{Tanggal (desimal)} = 6 + (17/30) = 6,567$$

Gunakanlah data di atas dengan memanfaatkan interpolasi polinomial untuk melakukan prediksi jumlah kasus baru Covid-19 pada tanggal-tanggal berikut:

- 16/07/2022
- 10/08/2022
- 05/09/2022
- Masukan user lainnya berupa tanggal (desimal) yang sudah diolah dengan asumsi prediksi selalu dilakukan untuk tahun 2022.

**Jawaban :**

The screenshot shows an IDE with three text files on the left, each containing 11 data points. The middle pane shows the source code of a Java class named `Polinom` with a `main` method that calls `polynomialInterpolation()`. The right pane shows the terminal output, which displays the polynomial equations for each file and the results of the interpolation at specific points.

```

test > 5b_a.txt
1 6.567 12624
2 7 21807
3 7.258 38391
4 7.451 54517
5 7.548 51952
6 7.839 28228
7 8.161 35764
8 8.484 20813
9 8.709 12408
10 9 10534
11 7.516

test > 5b_b.txt
1 6.567 12624
2 7 21807
3 7.258 38391
4 7.451 54517
5 7.548 51952
6 7.839 28228
7 8.161 35764
8 8.484 20813
9 8.709 12408
10 9 10534
11 8.322

test > 5b_c.txt
1 6.567 12624
2 7 21807
3 7.258 38391
4 7.451 54517
5 7.548 51952
6 7.839 28228
7 8.161 35764
8 8.484 20813
9 8.709 12408
10 9 10534
11 9.166

src > Polinom.java > Polinom
5 public class Polinom {
173     public static void main(String[] args) {
174         polynomialInterpolation();
175     }
176 }

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

2
Masukkan nama file teks: 5b_a
f(x) = (-141006.35265450666)x^9 + (9373741.525538526)x^8 + (-2.7550250306447875E8)x^7 + (4.696316966551336E9)x^6 + (-5.11378648581
3281E10)x^5 + (3.685975674434824E11)x^4 + (-1.757053341532463E12)x^3 + (5.335014967217977E12)x^2 + (-9.348572690402438E12)x^1 + 7.
188430348204094E12, f(7.516) = 53537.625
PS D:\Programs\Algeo01-23132\src> cd "d:\Programs\Algeo01-23132\src\" ; if ($?) { javac Polinom.java } ; if ($?) { java Polinom }
Apakah ingin ketik manual atau baca dari file?
1. Ketik manual
2. Baca dari file
2
Masukkan nama file teks: 5b_b
f(x) = (-141006.35265450666)x^9 + (9373741.525538526)x^8 + (-2.7550250306447875E8)x^7 + (4.696316966551336E9)x^6 + (-5.11378648581
3281E10)x^5 + (3.685975674434824E11)x^4 + (-1.757053341532463E12)x^3 + (5.335014967217977E12)x^2 + (-9.348572690402438E12)x^1 + 7.
188430348204094E12, f(8.322) = 36343.890625
PS D:\Programs\Algeo01-23132\src> cd "d:\Programs\Algeo01-23132\src\" ; if ($?) { javac Polinom.java } ; if ($?) { java Polinom }
Apakah ingin ketik manual atau baca dari file?
1. Ketik manual
2. Baca dari file
2
Masukkan nama file teks: 5b_c
f(x) = (-141006.35265450666)x^9 + (9373741.525538526)x^8 + (-2.7550250306447875E8)x^7 + (4.696316966551336E9)x^6 + (-5.11378648581
3281E10)x^5 + (3.685975674434824E11)x^4 + (-1.757053341532463E12)x^3 + (5.335014967217977E12)x^2 + (-9.348572690402438E12)x^1 + 7.
188430348204094E12, f(9.166) = -659035.09375

```

#### 4.5.3 Studi Kasus 12:

Sederhanakan fungsi  $f(x)$  yang memenuhi kondisi

$$f(x) = \frac{x^2 + \sqrt{x}}{e^x + x}$$

dengan polinom interpolasi derajat  $n$  di dalam selang  $[0, 2]$ .

Sebagai contoh, jika  $n = 5$ , maka titik-titik  $x$  yang diambil di dalam selang  $[0, 2]$  berjarak  $h = (2 - 0)/5 = 0.4$ .

**Jawaban :**

```

Masukkan banyak titik: 5
0.4 0.418884
0.8 0.507158
1.2 0.560925
1.6 0.583686
2 0.576652
1
f(x) = (-0.003727213541666442)x^4 + (0.024026041666665672)x^3 + (-0.1505880208333319)x^2 + (0.3780595833333256)x^1
+ 0.2903120000000001, f(1.0) = 0.5380823906250001

```

## 4.6 Regresi Linier

#### 4.6.1 Studi Kasus 14 :

Diberikan sekumpulan data sesuai pada tabel berikut ini.

Table 12.1: Data for Example 12.1

Nitrous Oxide, $y$	Humidity, $x_1$	Temp., $x_2$	Pressure, $x_3$	Nitrous Oxide, $y$	Humidity, $x_1$	Temp., $x_2$	Pressure, $x_3$
0.90	72.4	76.3	29.18	1.07	23.2	76.8	29.38
0.91	41.6	70.3	29.35	0.94	47.4	86.6	29.35
0.96	34.3	77.1	29.24	1.10	31.5	76.9	29.63
0.89	35.1	68.0	29.27	1.10	10.6	86.3	29.56
1.00	10.7	79.0	29.78	1.10	11.2	86.0	29.48
1.10	12.9	67.4	29.39	0.91	73.3	76.3	29.40
1.15	8.3	66.8	29.69	0.87	75.4	77.9	29.28
1.03	20.1	76.9	29.48	0.78	96.6	78.7	29.29
0.77	72.2	77.7	29.09	0.82	107.4	86.8	29.03
1.07	24.0	67.7	29.60	0.95	54.9	70.9	29.37

Source: Charles T. Hare, "Light-Duty Diesel Emission Correction Factors for Ambient Conditions," EPA-600/2-77-116. U.S. Environmental Protection Agency.

Gunakan *Normal Estimation Equation for Multiple Linear Regression* untuk mendapatkan regresi linear berganda dari data pada tabel di atas, kemudian estimasi nilai Nitrous Oxide apabila Humidity bernilai 50%, temperatur 76°F, dan tekanan udara sebesar 29.30.

Dari data-data tersebut, apabila diterapkan *Normal Estimation Equation for Multiple Linear Regression*, maka diperoleh sistem persamaan linear sebagai berikut.

$$20b_0 + 863.1b_1 + 1530.4b_2 + 587.84b_3 = 19.42$$

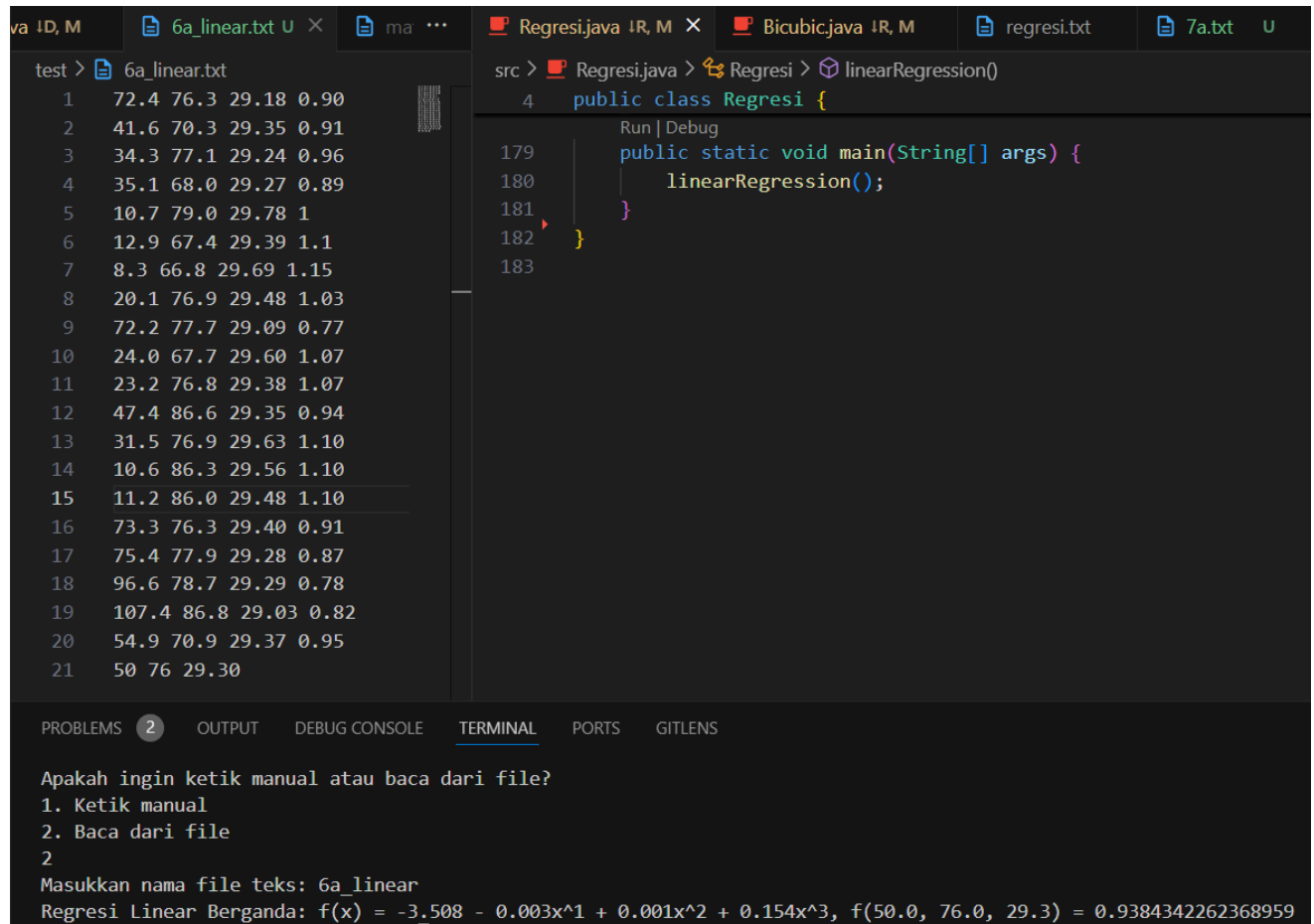
$$863.1b_0 + 54876.89b_1 + 67000.09b_2 + 25283.395b_3 = 779.477$$

$$1530.4b_0 + 67000.09b_1 + 117912.32b_2 + 44976.867b_3 = 1483.437$$

$$587.84b_0 + 25283.395b_1 + 44976.867b_2 + 17278.5086b_3 = 571.1219$$

Silahkan terapkan model-model ini pada *Multiple Quadratic Equation* juga dan bandingkan hasilnya. Sistem persamaan linear tidak akan diberikan untuk kasus ini.

**Jawaban :**



```

test > 6a_linear.txt
1 72.4 76.3 29.18 0.90
2 41.6 70.3 29.35 0.91
3 34.3 77.1 29.24 0.96
4 35.1 68.0 29.27 0.89
5 10.7 79.0 29.78 1
6 12.9 67.4 29.39 1.1
7 8.3 66.8 29.69 1.15
8 20.1 76.9 29.48 1.03
9 72.2 77.7 29.09 0.77
10 24.0 67.7 29.60 1.07
11 23.2 76.8 29.38 1.07
12 47.4 86.6 29.35 0.94
13 31.5 76.9 29.63 1.10
14 10.6 86.3 29.56 1.10
15 11.2 86.0 29.48 1.10
16 73.3 76.3 29.40 0.91
17 75.4 77.9 29.28 0.87
18 96.6 78.7 29.29 0.78
19 107.4 86.8 29.03 0.82
20 54.9 70.9 29.37 0.95
21 50 76 29.30

src > Regresi.java > Regresi > linearRegression()
4 public class Regresi {
    Run | Debug
    public static void main(String[] args) {
        linearRegression();
    }
179
180
181
182
183

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Apakah ingin ketik manual atau baca dari file?
1. Ketik manual
2. Baca dari file
2
Masukkan nama file teks: 6a_linear
Regresi Linear Berganda: f(x) = -3.508 - 0.003x^1 + 0.001x^2 + 0.154x^3, f(50.0, 76.0, 29.3) = 0.9384342262368959

```

## 4.7 Interpolasi Bicubic

### 4.7.1 Studi Kasus 13 :

Diberikan matriks input dengan bentuk sebagai berikut. Format matriks masukan bukan mewakili nilai matriks, tetapi mengikuti format masukan pada bagian “Spesifikasi Tugas” nomor 7.

$$\begin{pmatrix} 21 & 98 & 125 & 153 \\ 51 & 101 & 161 & 59 \\ 0 & 42 & 72 & 210 \\ 16 & 12 & 81 & 96 \end{pmatrix}$$

Tentukan nilai:

$$\begin{aligned} f(0, 0) &= ? \\ f(0.5, 0.5) &= ? \\ f(0.25, 0.75) &= ? \\ f(0.1, 0.9) &= ? \end{aligned}$$

**Jawaban :**

```
7a.txt  U x  matp ...  7b.txt  U x  ...  7c.txt  U x  ...  7d.txt  U x  ...  Bicubic.java  IR, M  X

test > 7a.txt
1 21 98 125 153
2 51 101 161 59
3 0 42 72 210
4 16 12 81 96
5 0 0

test > 7b.txt
1 21 98 125 153
2 51 101 161 59
3 0 42 72 210
4 16 12 81 96
5 0.5 0.5

test > 7c.txt
1 21 98 125 153
2 51 101 161 59
3 0 42 72 210
4 16 12 81 96
5 0.25 0.75

test > 7d.txt
1 21 98 125 153
2 51 101 161 59
3 0 42 72 210
4 16 12 81 96
5 0.1 0.9

src > Bicubic.java > Bicubic > main(String[])
8 public class Bicubic {
114     public static void main(String[] args) {
115         bicubicInterpolation();
116     }
117 }
118

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Masukkan nama file teks: 7a
Matriks :
21.0 98.0 125.0 153.0
51.0 101.0 161.0 59.0
0.0 42.0 72.0 210.0
16.0 12.0 81.0 96.0
f(0.0, 0.0) = 21.0
PS D:\Programs\Algeo01-23132\src> cd "d:\Programs\Algeo01-23132\src\" ; if ($?) { javac Bicubic.java } ; if ($?) { java Bicubic }
Masukkan nama file teks: 7b
51.0 101.0 161.0 59.0
0.0 42.0 72.0 210.0
16.0 12.0 81.0 96.0
f(0.5, 0.5) = 87.5
PS D:\Programs\Algeo01-23132\src> cd "d:\Programs\Algeo01-23132\src\" ; if ($?) { javac Bicubic.java } ; if ($?) { java Bicubic }
Masukkan nama file teks: 7c
Matriks :
21.0 98.0 125.0 153.0
51.0 101.0 161.0 59.0
0.0 42.0 72.0 210.0
16.0 12.0 81.0 96.0
f(0.25, 0.75) = 118.6220703125
PS D:\Programs\Algeo01-23132\src> cd "d:\Programs\Algeo01-23132\src\" ; if ($?) { javac Bicubic.java } ; if ($?) { java Bicubic }
Masukkan nama file teks: 7d
Matriks :
21.0 98.0 125.0 153.0
51.0 101.0 161.0 59.0
0.0 42.0 72.0 210.0
16.0 12.0 81.0 96.0
f(0.1, 0.9) = 129.025952
```



# Lampiran

Anton, H., & Rorres, C. (2013). *Elementary linear algebra: Applications Version* (11th ed.). Wiley.

Mulkek. (2020, January 1). *A unique solution, No solution, or Infinitely many solutions |  $Ax=b$*  [Video]. YouTube. [https://www.youtube.com/watch?v=18\\_oG-cD9Ck](https://www.youtube.com/watch?v=18_oG-cD9Ck)

Rupinder Sekhon and Roberta Bloom. (2022, July 18). *2.4: inverse matrices*. Mathematics LibreTexts.  
[https://math.libretexts.org/Bookshelves/Applied\\_Mathematics/Applied\\_Finite\\_Mathematics\\_\(Sekhon\\_and\\_Bloom\)/02%3A\\_Matrices/2.04%3A\\_Inverse\\_Matrices](https://math.libretexts.org/Bookshelves/Applied_Mathematics/Applied_Finite_Mathematics_(Sekhon_and_Bloom)/02%3A_Matrices/2.04%3A_Inverse_Matrices)

Lesson Explainer: Inverse of a Matrix: The Adjoint Method. (n.d.). Nagwa.  
<https://www.nagwa.com/en/explainers/403146289867/>

*3.2: Polynomial interpolation*. (2023, February 9). Mathematics LibreTexts.  
[https://math.libretexts.org/Courses/Angelo\\_State\\_University/Mathematical\\_Computing\\_with\\_Python/3%3A\\_Interpolation\\_and\\_Curve\\_Fitting/3.2%3A\\_Polynomial\\_Interpolation](https://math.libretexts.org/Courses/Angelo_State_University/Mathematical_Computing_with_Python/3%3A_Interpolation_and_Curve_Fitting/3.2%3A_Polynomial_Interpolation)

Fisher, Taylor. (2021, August 3). *Polynomial Interpolation*. GitHub Pages.  
<https://smashmath.github.io/math/polyinterp>

*Multivariate Polynomial Regression with Python*. (2024, January 31). Saturn Cloud.  
<https://saturncloud.io/blog/multivariate-polynomial-regression-with-python>

Video Presentasi :

[https://drive.google.com/drive/folders/1\\_e2SsFHG4qqS6g286IVn96L205GcCt?usp=drive\\_link](https://drive.google.com/drive/folders/1_e2SsFHG4qqS6g286IVn96L205GcCt?usp=drive_link)

Laporan (*up-to-date*):

<https://docs.google.com/document/d/1klkoLuk7ZxLK8diuO-OuKHSw0Q0as1nKhrKmHleGJlk>